FieldServer
FS-8700-47 DNP 3.0 Serial

Driver Manual

(Supplement to the FieldServer Instruction Manual)

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after November 2015

Kernel Version: 1.03
Document Revision: 14

Contact Information:

Thank you for purchasing the FieldServer.

Please call us for Technical support of the FieldServer product.

Contact Information:

Sierra Monitor Corporation
1991 Tarob Court
Milpitas, CA 95035

Contact number:
+1 408 262-6611
+1 800 727-4377

Email: info@sierramonitor.com

Website: www.sierramonitor.com

TABLE OF CONTENTS

LIST OF FIGURES

1 DNP 3.0 DRIVER DESCRIPTION

The DNP 3.0 Serial Driver allows the FieldServer to transfer data to and from devices over RS-232 or RS-485 using DNP 3.0 Driver protocol. The FieldServer can emulate either a Server or Client.

The following description of DNP is from the DNP User Group internet site.*

"The development of DNP was a comprehensive effort to achieve open, standards-based interoperability between substation computers, RTUs, IEDs (Intelligent Electronic Devices) and master stations (except inter-master station communications) for the electric utility industry. Also important was the time frame and the need for a solution to meet today's requirements. As ambitious an undertaking as this was, we are reaching this objective.

DNP is based on the standards of the International Electrotechnical Commission (IEC) Technical Committee 57, Working Group 03 who have been working on an OSI 3 layer "Enhanced Performance Architecture" (EPA) protocol standard for telecontrol applications. DNP has been designed to be as close to compliant as possible to the standards as they existed at time of development with the addition of functionality not identified in Europe but needed for current and future North American applications (e.g. limited transport layer functions to support 2K descriptor transfers for IEDs, RF and fiber support). Recently DNP 3.0 was selected as a Recommended Practice by the IEEE C.2 Task Force; RTU to IED Communications Protocol.

Feature Rich

DNP offers flexibility and functionality that go far beyond conventional communications protocols. Among its robust and flexible features DNP 3.0 includes:

- *Output options*
- *Secure configuration/file transfers*
- *Addressing for over 65,000 devices on a single link*
- *Time synchronization and time-stamped events*
- *Broadcast messages*
- *Data link and application layer confirmation*

DNP 3.0 was originally designed based on three layers of the OSI seven-layer model: application layer, data link layer and physical layer. The application layer is object-based with objects provided for most generic data formats. The data link layer provides for several methods of retrieving data such as polling for classes and object variations. The physical layer defines most commonly a simple RS-232 or RS-485 interface.

DNP 3.0 is very efficient for a layered protocol while ensuring high data integrity.

Suits Any SCADA/EMS Environment

Because DNP 3.0 is based on the IEC 870-5 requirements, DNP is suitable for application in the entire SCADA/EMS environment. This includes RTU to IED communications, master to remote communications, and even peer-to-peer instances and network applications.

* DNP Users Group, PO Box 43075 DVPO, Calgary, AB, Canada T2J 7A7

Being an object-based application layer protocol, DNP 3.0 has the flexibility to support multiple operating modes such as poll-response, polled report-by-exception, unsolicited responses and peer-to-peer. It permits multiple masters and encourages distributed intelligence.

Users can expect many benefits from using DNP. In the short term:

- *Interoperability between multi-vendor devices*
- *Fewer protocols to support in the field*
- *reduced software costs*
- *No protocol translators needed*
- *Shorter delivery schedules*
- *Less testing, maintenance and training*
- *Improved documentation*
- *Independent conformance testing*
- *Support by independent users group and third-party sources (e.g. test sets, source code).*

2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by Sierra Monitor Corporation for this driver

Sierra Monitor Corporation PART #	Description
FS-8915-10	UTP cable (7 foot) for RS-232 use
FS-8917-04	RJ45 to DB25M connection adapter
FS-8700-47	Driver Manual.

3 HARDWARE CONNECTIONS

The FieldServer is connected to the DNP-3.0 device as shown below.

Configure the DNP-3.0 device according to manufacturer's instructions.

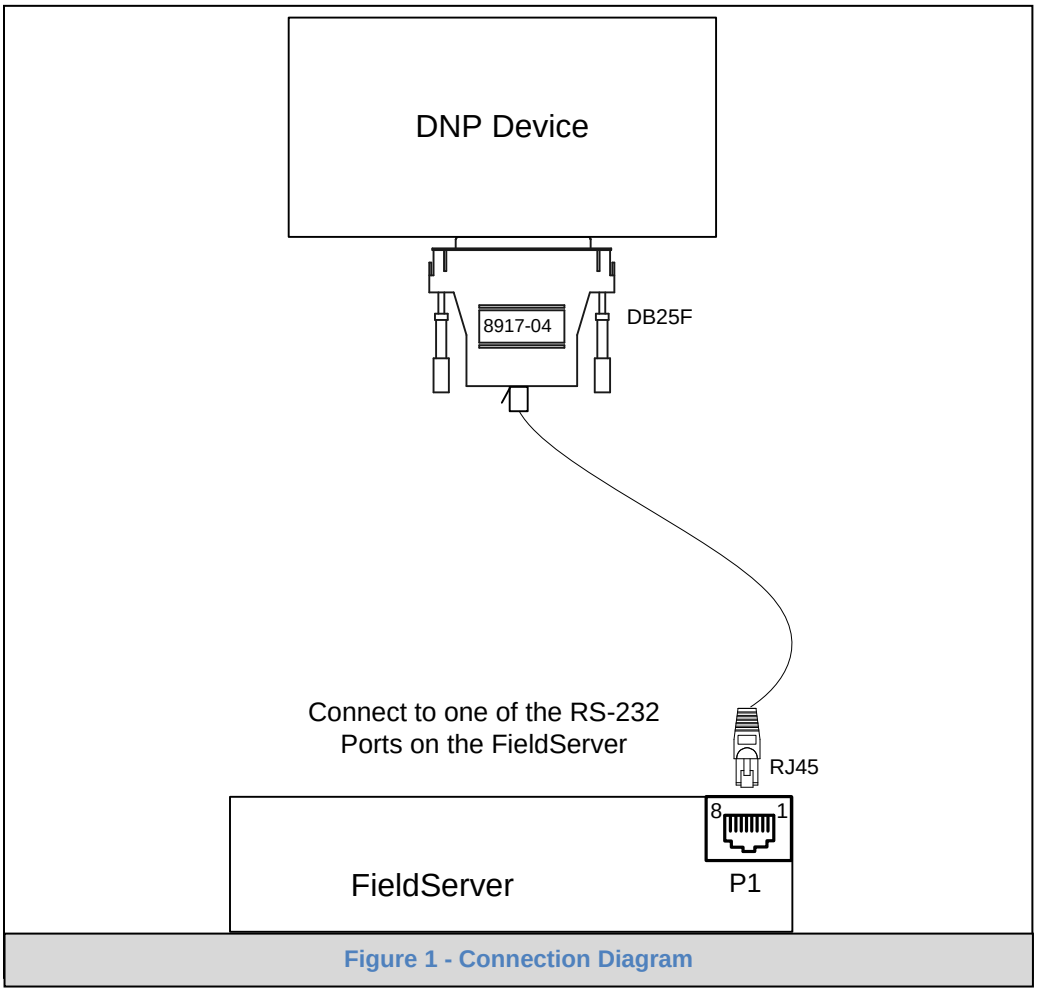


Figure 1 - Connection Diagram

FieldServer Function	From	Default	Color
Rx	RJ45-01	DB25F-02	White
CTS	RJ45-02	DB25F-04	Brown
DSR	RJ45-03		Yellow
GND	RJ45-04	DB25F-07	Green
GND	RJ45-05		Red
DTR	RJ45-06		Black
RTS	RJ45-07	DB25F-05	Orange
Tx	RJ45-08	DB25F-03	Blue

3.1 Connection Notes

Pinouts and adapters may vary according to the device being connected to. Refer to DNP installation manual for pin connection reference.

4 CONFIGURING THE FIELDSERVER AS A DNP 3.0 DRIVER CLIENT

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" files supplied with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a DNP 3.0 Driver Server.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for DNP 3.0 Driver communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the Servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the **bold** legal value being the default.

4.1 Data Arrays

Section Title		
Data_Arrays		
Column Title	**Function**	**Legal Values**
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Format	Provide data format. Each data array can only take on one format.	FLOAT, BIT, UInt16, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required for the data being placed in this array.	1-10,000

Example

```
//  Data Arrays
//
Data_Arrays
Data_Array_Name    , Data_Format    , Data_Array_Length
DA_AI_01           , UInt16         , 200
DA_AO_01           , UInt16         , 200
DA_DI_01           , Bit            , 200
DA_DO_01           , Bit            , 200
```

4.2 Client Side Connection Parameters

Section Title
Connections

Column Title	Function	Legal Values
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[1]
Baud*	Specify baud rate	110 – 115200, standard baud rates only
Parity*	Specify parity	Even, Odd, None, Mark, Space
Data_Bits*	Specify data bits	7, 8
Stop_Bits*	Specify stop bits	1
Protocol	Specify protocol used	DNP
Poll Delay*	Time between internal polls	0-32000 seconds, 1 second.
Application*	Versions of the driver prior to 1.02a used a different method to calculate DA offset. Refer to Appendix A.16. It is also possible to use this parameter to control if link resets are used/required. Refer to Appendix A.15.	OriginalStyle, NoLink, OrigStyle-NoLink

Example

```
//  Client Side Connections

Connections
Port        , Baud    , Parity    , Protocol    , Poll_Delay
R1          , 9600    , None      , DNP         , 0.100s
```

[1] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

4.3 Client Side Node Parameters

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for Node	Up to 32 alphanumeric characters
Node_ID	DNP 3.0 Station address of physical Server node	0-65535
Protocol	Specify Protocol used	DNP
Port	Specify through which port the device is connected to the FieldServer	P1-P8, R1-R2[2]

Example

```
//  Client Side Nodes
Nodes
Node_Name      , Node_ID     , Protocol    , Port
PLC 1          , 1           , DNP         , P1
```

4.4 Client Side Map Descriptor Parameters

4.4.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section **Error! Reference source not found.**.
Data_Array_Offset	Starting location in Data Array	0 to (Data_Array_Length-1) as specified in Section **Error! Reference source not found.**.
Function	Function of Client Map Descriptor	Rdbc, Wrbc, Wrbx

4.4.2 Driver Related Map Descriptor Parameters

Column Title	Function	Legal Values
The following parameters are used by a number of drivers.		
Node_Name	Name of Node to fetch data from	A Node Name specified in "Client Node Descriptor". Special Map Descriptors are discussed in **Error! Reference source not found.**.

2 Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

Column Title	Function	Legal Values
Length	Length of Map Descriptor. If a request length is too large the DNP 3.0 driver will produce a message and a panic. The maximum length is a function of the data object and data variation.	1 – 1000
Address	Starting address of data element to be read	0, 1 , 2 etc
The following parameters apply only to the DNP 3.0 Driver		
DnpDataType	Corresponds to the Data Object Types defined in the DNP data object Library. Additional information is provided in Appendix A.9	1, 2, 10, 12, 30, 31, 32, 33, 40, 41, 20, 22, 23, 50, 51, 52, 60, 80 - decimal numbers
DnpDataVari	Corresponds to the Data Object Variant defined in the DNP data object Library. Enter as decimal number. Additional information is provided in Appendix A.9	0, 1,2,3 etc Legal values are determined by the value of dnpDataType.
DnpSubType*	Used to tell driver which Suffield of the object to map to/from the FieldServer Data Array. Additional information is provided in 0	Value, flags, time1, time2, combo
DnpFlagBit*	Not Used.	
DnpQualifier*	This parameter is only required if you need to over-ride the default qualifier used by the DNP 3.0 driver. Refer to Appendix A.11	Zero, 1, 6, 7, 8, 17h - hexadecimal values. For qualifier 17h specify the value of dnpQualifier as 17 in the Map Descriptor. For Qualifier zero use the string “zero”
DnpFunction*	This parameter is only required if you need to over-ride the default function used by the DNP 3.0 driver. Refer to 0	Legal DNP function codes. Correspond to the function code required on vendor’s implementation table.
DnpAssociate*	When class data is requested the DNP device responds with data of multiple types and variations in one message. One Map Descriptor is used per data type - this parameter is used to link these Map Descriptors.	Non-zero positive integers.
DnpMultiMsg*	This parameter is used to produce a single message with a request for multiple object types. Assign positive whole numbers to associate Map Descriptors for this purpose. All Map Descriptors whose dnpMultiMsg values are equal will be requested in a single poll. Ensure only one is active (rdbc for example) and all the others have the function set to 'Server'. Refer also to Appendix A.20	0, positive whole numbers. By default Map Descriptors are not associated with each other. The default value of zero ensures no association.

4.4.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	≥0.1s

4.4.4 Map Descriptor Example.

In this example we read 10 Analog Inputs from a DNP device. The request is with respect to inputs 20 to 29 and we will place the results of the read in the data array called DA_AI3 in positions 20 to 29. In this example the dnpSubType is not specified so the driver will report the value of the object.

```
//  Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   ,Data_Array_Name   ,Data_Array_Offset   ,Function   ,Node_Name   ,Address   ,Length   ,Scan_Interval   ,dnpDataType   ,dnpDataVari
A1                    ,DA_AI3           ,20                  ,Rdbc       ,Node_A      ,20         ,10       ,5               ,30            ,1
```

Any useful name. Duplicates are allowed but may make some of the validation messages less meaningful.

The results of the read are placed in this data array.

The value of the first analog input read will be placed in position 20 of the data table (the 21st element of the table.)

This Map Descriptor will perform a read of the DNP device and the read will be performed continuously.

This is the name of the node that we are reading. This name must connect to a node definition in the CSV file which will contain the address of the device being read and this in turn should connect to a port definition though which the FieldServer will communicate with the DNP device.

Read Analog Inputs in the slave device starting at input index 20. Vendor documentation will tell you the indices of the inputs you are interested in.

DNP Object Variation is *16 Bit Analog Input*

DNP Object Type is *Analog Input*

Read 10 elements.

4.4.5 Map Descriptor Example 2 – Simple Read – Extract the Quality/Status

In this example the status data rather than the input values are written to the data array specified.

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Address ,Length ,Scan_ Interval ,dnpData Type ,dnpDataVari ,dnpSub Type
A1                  ,DA_AI3STAT      ,20               ,Rdbc     ,Node_A    ,20       ,10      ,5             ,30           ,1           Flags
```

The results of the read are placed in this data array.

DNP Object Type is *Analog Input*

DNP Object Variation is *16 Bit Analog Input*

Tells the driver to write the status byte to the data array, ignoring the value of the object being read. You can use *value* / *time1* / *time2* / *flags* / *combo* as legal values for this parameter. The default is *value*.

4.4.6 Map Descriptor Example 3 – Simple Read Extract Value & Status

In this example we extract both the value and the status using two Map Descriptors.

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Address ,Length ,Scan_Interval ,dnpDataType ,dnpDataVari ,dnpSubType ,dnpAsscoiate
A1                  ,DA_AI3          ,20               ,Rdbc     ,Node_A    ,20       ,10      ,5             ,30           ,1           ,Value      ,1
A2                  ,DA_AI3STAT      ,20               ,Passive  ,Node_A    ,20       ,10      ,5             ,30           ,1           ,Flags      ,1
```

The status data is placed into a different array from the value data.

Only one Map Descriptor need be active. The active Map Descriptor generates the poll. Both the active and passive Map Descriptors are used to process the response.

The common value for dnpAssociate ties these two Map Descriptors together and tells the driver to use them both to process the response data.

4.4.1 Map Descriptor Example 4 – Reading Class Data

In this example we read class data. A class is a structure of different data types and variations. Vendors group data into classes based on the relationships and importance of the data. In this example you will see that only one of the Map Descriptors is active – the class data read. The remaining passive Map Descriptors are used to process the response from the class data read. The response may consist of a number of data object types and variations and is dependent on the vendor's implementation of DNP. Some vendors allow users to configure what objects constitute a class.

This Map Descriptor requests class 1 data from the DNP device.

```
// Class 1 Data
Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Address ,Length ,dnpSubType ,dnpDataType ,dnpDataVari ,dnpQualifier ,dnpAssociate ,timeout
Class1-Data ,13.0s ,DA_A3 ,0 ,Rdbc ,Node_A ,0 ,100 ,value ,60 ,1 ,6 ,1 ,5.0s
Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Address ,Length ,dnpSubType ,dnpDataType ,dnpDataVari ,dnpAssociate
Class1-di ,3.0s ,CL_1 ,0 ,Passive ,Node_A ,0 ,156 ,value ,1 ,0 ,1
Class1-do ,3.0s ,CL_2 ,0 ,Passive ,Node_A ,0 ,9 ,value ,10 ,0 ,1
Class1-co ,3.0s ,CL_3 ,0 ,Passive , Node_A ,0 ,39 ,value ,20 ,0 ,1
Class1-ai-1 ,3.0s ,CL_4 ,0 ,Passive ,Node_A ,0 ,100 ,value ,30 ,0 ,1
Class1-ai-2 ,3.0s ,CL_5 ,0 ,Passive ,Node_A ,100 ,100 ,value ,30 ,0 ,1
Class1-ai-3 ,3.0s ,CL_6 ,0 ,Passive ,Node_A ,200 ,100 ,value ,30 ,0 ,1
```

These Map Descriptors process the data that the device returns. If more data is returned that you have defined Map Descriptors for then it will be abandoned.

The common value for dnpAssociate ties these Map Descriptors together and tells the driver to use them both to process the response data.

4.4.2 Map Descriptor Example 5 – Read an unknown quantity of points (Qualifier 6)

In this example the qualifier has been set to 6. This is the qualifier the protocol uses to request all possible points of a particular object type and variation. Ensure that there is sufficient length to store all possible data.

```
//  Client Side Map Descriptors
Map-Descriptors
Map_Descriptor_Name  ,Data_Array_Name  ,Data_Array_Offset  ,Function  ,Node_Name  ,Address  ,Length  ,Scan_Interval  ,dnpDataType  ,dnpDataVari  ,DnpQualifier
ReadAll_AI's         ,DA_AI            ,0                 ,Rdbc      ,Node_A     ,0         ,100      ,5.0s           ,30           ,1            6
```

Enough space has been reserved for 100 objects. If the response contains more then there will be an error.

The Qualifier of 6 tells the driver to request all data objects.

4.4.3 Map Descriptor Example 6 – Read an unknown variation

If the DNP device does not document the object type variation that it serves, set the variation to zero. The remote device will respond with its default variation. The problem is that the response cannot be stored using this Map Descriptor because the response will have its variation non-zero. Thus you would need to make a Server Map Descriptor for each possible variation. A better strategy is to experiment. Use the single Map Descriptor below, wait for the error and then modify the CSV based on information found in the error message.

```
//  Client Side Map Descriptors
Map_Descriptors
Map_Descriptor_Name  ,Data_Array_Name  ,Data_Array_Offset  ,Function  ,Node_Name  ,Address  ,Length  ,Scan_Interval  ,dnpDataType  ,dnpDataVari  ,DnpQualifier
ReadAll_AI's         ,DA_AI            ,0                 ,Rdbc      ,Node_A     ,0         ,100      ,5.0s           ,30           ,0            ,6
```

With the variation set to zero the remote device will respond with its default variation.

5 CONFIGURING THE FIELDSERVER AS A DNP 3.0 DRIVER SERVER

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" files provided with the FieldServer.)
This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a DNP 3.0 Driver Client.
The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for DNP 3.0 Driver communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the FieldServer virtual node(s) needs to be declared in the "Server Side Nodes" section, and the data to be provided to the Clients needs to be mapped in the "Server Side Map Descriptors" section. Details on how to do this can be found below.
Note that in the tables, * indicates an optional parameter, with the **bold** legal value being the default.

5.1 Server Side Connection Descriptors

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[3]
Baud*	Specify baud rate	110 – 115200, standard baud
Parity*	Specify parity	Even, Odd, None, Mark, Space
Data_Bits*	Specify data bits	7, 8
Stop_Bits*	Specify stop bits	1
Protocol	Specify protocol used	DNP

Example

```
//  Server Side Connections
Connections
Port                    ,Baud   ,Parity   ,Protocol
P8                      ,9600   ,None     ,DNP
```

[3] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

5.2 Server Side Node Descriptors

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	DNP 3.0 station address of physical Server node	0-65535
Protocol	Specify protocol used	DNP
Class_Data_Serving_Ctr*l	This parameter can be specified to configure the Server to serve changed data only. Refer to Example 5.3.8 for more information.	Class0; Class1; Class2; Class3; Static
Server_II_Array	The name of a Data Array that has previously been defined in the configuration in the Data Arrays section. Refer to Section 4.1.	Max 15 characters

Example

```
//    Server Side Nodes
Nodes
Node_Name        ,Node_ID   ,Protocol
FieldServer      ,11        ,DNP
```

5.3 Server Side Map Descriptors

5.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from “Data Array” section above
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in “Data Array” section above
Function	Function of Client Map Descriptor	Generally for Server side nodes you will use the PASSIVE function. The WRBX function may be used to generate unsolicited messages.

5.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	A Node Name specified in "Client Node Descriptor". Special Map Descriptor's used by the DNP 3.0 Driver are discussed in Section 4.4
Length	Length of Map Descriptor. If a request length is too large the DNP 3.0 driver will produce a message and a panic. The maximum length is a function of the data object and data variation being processed.	1 – 1000
Address	Starting address of data element to be read	0, 1 , 2 etc
The following parameters apply only to the DNP 3.0 Driver		
DnpDataType	Corresponds to the Data Object Types defined in the DNP data object Library. Enter as decimal number. Refer to Appendix A.9	1, 2, 10, 12, 30, 31, 32, 33, 40, 41, 20, 22, 23, 50,51, 52, 60, 80
DnpDataVari	Corresponds to the Data Object Variant defined in the DNP data object Library. Enter as decimal number. When configured as a Server the driver can respond to requests for the so called 'Default' variation. These are polls where the variation is zero. To configure the driver to be able to respond to requests for the default variation then you must create a MapDesc where the DNPDataVari=0. Note that the driver considers variation=1 as the default in most case.	0,1,2,3 etc Legal values are determined by the value of dnpDataType. Additional information is provided in Appendix A.9
DnpQualifier*	This parameter is ignored by the driver acting as a Server. The qualifier of the incoming poll is used to form the response. If the poll qualifier is not supported by the driver's response function then the driver responds with Qualifier 1. The response function supports the following qualifiers; 0,1,6,7,8,17,28,	
DnpMultiMsg	Simply ensure that that there is a Server MD for each object requested. No special actions are required to configure the Server to respond to requests for multiple object types. Refer to Appendix A.20 for more information.	

5.3.3 Timing Parameters

Column Title	Function	Legal Values
Scada_Hold_Timeout	Time Server side waits before notifying Client that node is offline on FieldServer Client side.	>1.0s

5.3.4 Map Descriptor Example 1 – Discrete Data.

In this example the following Map Descriptor will be used allow a remote device to request discrete input data.

```
//    Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   ,Data_Array_Name   ,Data_Array_Offset   ,Function   ,Node_Name   ,Address   ,Length   ,Scada_Hold_Timeout   ,dnpDataType   ,dnpDataVari
A1                    ,DA_DI1            ,0                   ,Passive    ,Node_A      ,0         ,10        ,5                   ,1             ,1
```

The DNP 3.0 driver will use data from this array to form the response.

This Map Descriptor can respond to a read/write poll from a remote DNP device.

This Map Descriptor will be used to process any poll whose objects match and whose addresses fall inside the address and length range specified by this Map Descriptor.

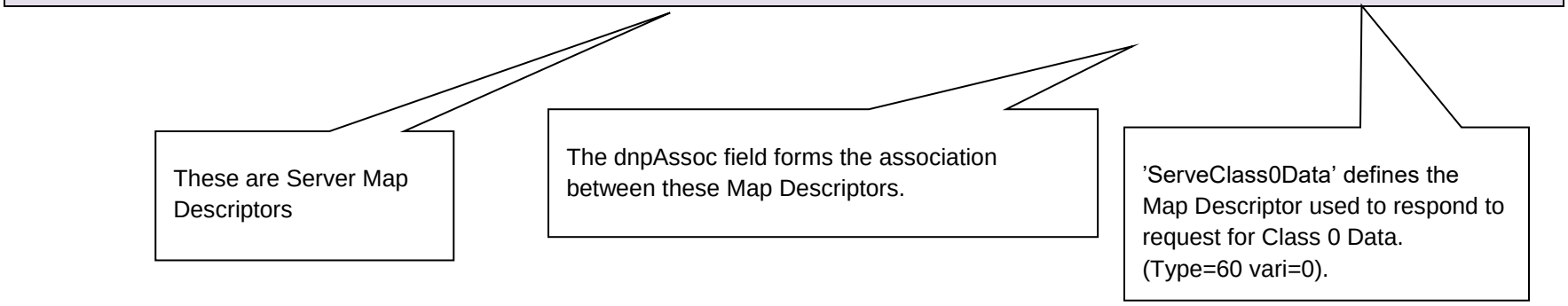
This is Map Descriptor will process commands for a DNP Object Type 1 variation 1 Map Descriptor i.e. binary input.

5.3.5 Map Descriptor Example 2 – Class Data

Class data is a composite set of data comprising data points from different data types. In this example the Map Descriptors have been created to serve class 0 data. Map Descriptors associated with the Class 0 Map Descriptor tell the driver which data to serve as the class data. When a request is received it is matched against a Class Map Descriptor. If a matching Map Descriptor is found the driver builds a response using the data requested in the associated Map Descriptors. In this example 10 Digital Inputs and 2 Analog Inputs will be served as the response.

```
//    Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name          ,Data_Array_Name  ,Data_Array_Offset  ,Function  ,Node_Name  ,Address  ,Length  ,DnpAssoc  ,dnpDataType  ,dnpDataVari
ServeClass0Data              ,DA_Dummy         ,0                 ,Server    ,Node_A     ,0        ,1        ,1         ,60           ,0
ServeClass0Data_DI  ,DA_DI   ,0                ,Server            ,Node_A    ,0          ,10       ,1         ,1         ,1            ,1
ServeClass0Data_AI  ,DA_AI   ,0                ,Server            ,Node_A    ,0          ,2        ,1         ,1         ,30           ,1
```



5.3.6 Map Descriptor Example 3 – Class Data (another example)

In this example the Server responds with a number of non-consecutive binary points. All will work fine except when the Client polls for this data with qualifier 7. The default response will be qualifier 7 with implied index style 0. The problem is that qualifier 7 tells the Client the number of objects in the packet, but they are all assumed to be addressed sequentially from address 0. When the data is served, the Client will unpack each portion, thinking that the address is zero for each portion and hence the data will be overwritten. To prevent this add the dnpIndexStyle parameter with a value of 1 to force the Server to prefix each point of data with the address of the point and thus allow the Client to unpack the data correctly. All this can be avoided by having the Client poll for data with a qualifier of 1.

```
//    Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name  ,Data_Array_Name  ,Data_Array_Location  ,Function  ,Node_Name  ,Address  ,Length  ,DNPDataType  ,DNPDataVari  ,DNPAssociate
Class_0_Data         ,DA_DUMMY         ,0                   ,Server    ,RTU        ,0         ,1        ,60           ,0
Map_Descriptors
Map_Descriptor_Name  ,Data_Array_Name  ,Data_Array_Location  ,Function  ,Node_Name  ,Address  ,Length  ,DNPDataType  ,DNPDataVari  ,DNPAssociate  ,dnpIndexStyle
Class_0_Analogs      ,DA_AI_01         ,0                   ,Server    ,RTU        ,1         ,16       ,32           ,1            ,1            ,1
Class_0_Discs_1      ,DA_DI_01         ,0                   ,Server    ,RTU        ,1         ,3        ,2            ,1            ,1            ,1
Class_0_Discs_2      ,DA_PB_01         ,5                   ,Server    ,RTU        ,4         ,1        ,2            ,1            ,1            ,1
Class_0_Discs_3      ,DA_PB_01         ,7                   ,Server    ,RTU        ,5         ,1        ,2            ,1            ,1            ,1
Class_0_Discs_4      ,DA_PB_01         ,9                   ,Server    ,RTU        ,6         ,1        ,2            ,1            ,1            ,1
Class_0_Discs_5      ,DA_PB_01         ,11                  ,Passive   ,RTU        ,7         ,1        ,2            ,1            ,1            ,1
Class_0_Discs_6      ,DA_PB_01         ,13                  ,Passive   ,RTU        ,8         ,1        ,2            ,1            ,1            ,1
Class_0_Discs_7      ,DA_PB_01         ,15                  ,Passive   ,RTU        ,9         ,1        ,2            ,1            ,1            ,1
Class_0_Discs_8      ,DA_PB_01         ,17                  ,Passive   ,RTU        ,10        ,1        ,2            ,1            ,1            ,1
Class_0_Discs_9      ,DA_PB_01         ,27                  ,Passive   ,RTU        ,11        ,1        ,2            ,1            ,1            ,1
Class_0_Discs_10     ,DA_PB_01         ,29                  ,Passive   ,RTU        ,12        ,1        ,2            ,1            ,1            ,1
Class_0_Discs_11     ,DA_PB_01         ,41                  ,Passive   ,RTU        ,13        ,1        ,2            ,1            ,1            ,1
Class_0_Discs_12     ,DA_PB_01         ,43                  ,Passive   ,RTU        ,14        ,1        ,2            ,1            ,1            ,1
Class_0_Discs_13     ,DA_PB_01         ,45                  ,Passive   ,RTU        ,15        ,1        ,2            ,1            ,1            ,1
Class_0_Discs_14     ,DA_PB_01         ,48                  ,Passive   ,RTU        ,16        ,1        ,2            ,1            ,1            ,1
```

```
Class_0_Discs_15    ,DA_PB_01    ,49    ,Passive    ,RTU    ,17    ,1    ,2    ,1    ,1    ,1
Class_0_Discs_16    ,DA_PB_01    ,51    ,Passive    ,RTU    ,18    ,1    ,2    ,1    ,1    ,1
Class_0_Discs_17    ,DA_PB_01    ,53    ,Passive    ,RTU    ,19    ,1    ,2    ,1    ,1    ,1
Class_0_Discs_18    ,DA_PB_01    ,55    ,Passive    ,RTU    ,20    ,1    ,2    ,1    ,1    ,1
Class_0_Discs_19    ,DA_PB_01    ,61    ,Passive    ,RTU    ,23    ,1    ,2    ,1    ,1    ,1
Class_0_Discs_2     ,DA_PB_01    ,63    ,Passive    ,RTU    ,24    ,1    ,2    ,1    ,1    ,1
```

5.3.7 Map Descriptor Example 4 – Default Variation (Variation zero)

If a remote Client polls for data of a particular type and the variation is set to zero then the request is for the Server's default variation. For the FieldServer, the default variation is always 1. This means that requests for variation zero will fail unless the Server contains a Map Descriptor where the variation is 1.

```
//    Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Na    ,Data_Array_Na    ,Data_Array_Off    ,Functio    ,Node_Na    ,Addres    ,Lengt    ,DnpAss    ,dnpDataTy    ,dnpDataV
ServeData_DI         ,DA_DI            ,0                 ,Server     ,Node_A     ,0         ,10       ,1         ,1            ,1
ServeData_AI         ,DA_AI            ,0                 ,Server     ,Node_A     ,0         ,2        ,1         ,30           ,1
```

With Data Objects of variation 1 defined the Server can respond to requests for variation zero. Such requests are polls for the default variation. The default is 1 for this driver.

5.3.8 Map Descriptor Example 5 - Class_Data_Serving_Ctrl

To configure the Server to serve changed data only, specify this parameter for the node of interest. If the parameter value = Classx and Classx data is requested by the remote client, the driver will use the subscriptions configured for the data objects that constitute the class to determine which data objects should be served and will only serve them if they have been changed and the change meets the subscription requirements.

If the parameter is not specified or the value is specified as static then all data objects are served, irrespective of whether they have changed or not.

In this example Class 0 has been configured to serve changed data only. The parameter 'Class_Data_Serving_Ctrl' has been allocated a value of 'Class0'. All other classes will serve the complete set of data objects that define the class. When Class 0 data is requested the driver will do the following:

- Check each data object that forms the class to see whether it has been updated by the downstream protocol.
- If it has been updated since it was last served then its value is checked against the data objects subscription 'COV_DeadBand'.
- If the value has changed by at least that much then the data object is served. If a subscription has not been defined for the point then it will not be served.

When the data is served the objects may be non-consecutive since the Server serves only 'changed' data. In such cases the Server may change the qualifier of the response from the default or configured qualifier because the points being served in the response may not be sequential. Qualifier 17 will be used.

```
// Server Side Nodes
Node_Name          ,Node_ID  ,Protocol  ,Class_Data_Serving_Ctrl  Port
Node_DNP3          ,1        ,DNP       ,Class0                  ,R1
```

The Class Map Descriptor is a dummy, used only to match an incoming request and find “Associated” Map Descriptor's which define the composite data set that must be served in the response

Map_Descriptors									
Map_Descriptor_Name	,Data_Array_Name	,Data_Array_Offset	,Function	,Node_Name	,Address	,Length	,dnpDataType	,dnpDataVari	,dnpAssociate
Class_0	,Class_DA0	,0	,Server	,Node_DNP3	,0	,1	,60	,1	,1
Class_0_Bin	,Bin_Inputs	,0	,Server	,Node_DNP3	,0	,20	,2	,2	,1
Class_0_Ana	,Ana_Inputs	,0	,Server	,Node_DNP3	,0	,7	,32	,4	,1
Class_1	,Class_DA1	,0	,Server	,Node_DNP3	,0	,1	,60	,2	,2
Class_1_Bin	,Bin_Inputs	,0	,Server	,Node_DNP3	,0	,20	,1	,1	,2
Class_1_Ana	,Ana_Inputs	,0	,Server	,Node_DNP3	,0	,7	,30	,1	,2
Class_2	,Class_DA2	,0	,Server	,Node_DNP3	,0	,1	,60	,3	,3
Class_2_Bin	,Bin_Inputs	,0	,Server	,Node_DNP3	,0	,20	,1	,1	,3
Class_3	,Class_DA3	,0	,Server	,Node_DNP3	,0	,1	,60	,4	,4
Class_3_Ana	,Ana_Inputs	,0	,Server	,Node_DNP3	,0	,7	,30	,1	,4

Subscriptions			
Data_Array_Name	,Data_Array_Offset	,Node_Name	,COV_Deadband
Bin_Inputs	,0	,Node_DNP3	,1
Bin_Inputs	,1	,Node_DNP3	,1
Bin_Inputs	,2	,Node_DNP3	,1
Bin_Inputs	,3	,Node_DNP3	,1
Bin_Inputs	,4	,Node_DNP3	,1
Bin_Inputs	,5	,Node_DNP3	,1
Bin_Inputs	,6	,Node_DNP3	,1
Bin_Inputs	,7	,Node_DNP3	,1
Bin_Inputs	,8	,Node_DNP3	,1
Bin_Inputs	,9	,Node_DNP3	,1
Bin_Inputs	,10	,Node_DNP3	,1
Bin_Inputs	,11	,Node_DNP3	,1
Bin_Inputs	,12	,Node_DNP3	,1
Bin_Inputs	,13	,Node_DNP3	,1
Bin_Inputs	,14	,Node_DNP3	,1

```
Bin_Inputs      ,15     ,Node_DNP3   ,1
Bin_Inputs      ,16     ,Node_DNP3   ,1
Bin_Inputs      ,17     ,Node_DNP3   ,1
Bin_Inputs      ,18     ,Node_DNP3   ,1
Bin_Inputs      ,19     ,Node_DNP3   ,1
Ana_Inputs      ,0      ,Node_DNP3   ,1.0
Ana_Inputs      ,1      ,Node_DNP3   ,1.0
Ana_Inputs      ,2      ,Node_DNP3   ,25.0
Ana_Inputs      ,3      ,Node_DNP3   ,65.0
Ana_Inputs      ,4      ,Node_DNP3   ,25.0
Ana_Inputs      ,5      ,Node_DNP3   ,25.0
Ana_Inputs      ,6      ,Node_DNP3   ,20.0
```

5.3.9 Map Descriptor Example 5 – Serving Objects which have a time field

Some Data Objects have a time field associated with them, e.g. Frozen Analog Input with Time of Freeze (Object = 31, Variation 3). When this object is served the Server must provide the value of the frozen input as well as the time of the freeze. The driver extracts the object's value from the Primary Data Array. The object's time is extracted from a secondary Data Array. If a secondary Data Array is not specified, the driver serves the current FieldServer time in UTC (Universal Coordinated Time). If a secondary Data Array is specified, the driver extracts the value and multiplies it by 1000 before serving it. Thus the Secondary Data Array is assumed to contain the number of seconds since Jan 1 1970 and not the number of milliseconds.

```
//    Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_N  ,Data_Array_Na  ,Data_Array_Off  ,Functi  ,Node_Na  ,Addre  ,Lengt  ,dnpDataTy  ,dnpDataV  ,DA_Byte_Na
Serve_FrozenAI_v  ,DA_AI          ,0               ,Server  ,Node_A   ,0       ,100     ,31         ,3         ,DA_AI_TIME
```

5.3.10 Map Descriptor Example 6 – Unsolicited Messages

A driver configured as an active Server can send unsolicited messages (not responses to a poll) to a remote Client when data changes.

In this example, 100 elements of the Data Array called DA_AI are monitored. If the data in any one of these elements is updated (even if the value doesn't change) then the driver generates a once off write message to the remote node called Node_A.

```
Subscriptions
Data_Array_Name   ,Data_Array_Offset   ,Node_Name   ,COV_Deadband
DA_Class0_bin     ,1                  ,Node_A       ,1
DA_Class0_bin     ,3                  ,Node_A       ,1
DA_Class0_ana     ,0                  ,Node_A       ,2.0
```

```
//   Client Side Map Descriptors
Map_Descriptors
Map_Descriptor_Na  ,Data_Array_Na  ,Data_Array_Off  ,Functio  ,Node_Na  ,Addres  ,Lengt  ,dnpDataTy  ,dnpDataV  ,DnpFuncti
EventMsg1          ,DA_AI          ,0               ,Wrbx      ,Node_A    ,0       ,100    ,30         ,0         ,130
```

5.4 Server Side Limitations

The DNP 3.0 Server can only parse a single poll per message. This means that a single message cannot contain more than one read request - You cannot read two different objects types/variations in a single read request. The same limitation applies to write commands sent the Server.

Appendix A. ADVANCED TOPICS

Appendix A.1. DNP 3.0 Protocol.

The DNP 3.0 protocol is complex and not all the features are implemented by this driver.

- The application layer performs a large set of potential functions, each of which can request its own app layer confirmation transaction and many of which include a separate response transaction.
- The app layer messages are wrapped and unwrapped by the data link layer which can ask for DLL layer ack's and confirmations.
- The protocol provides for unsolicited messages.
- The protocol defines and allows a huge set of data object & variations to be handled.
- Not all DNP devices (slaves) provide all functions, data objects....

Appendix A.2. DNP Driver Functionality

The DNP master driver has been developed to provide the functionality a FieldServer Technologies Client requires in communicating with a DNP slave device as well as additional functionality and data object handling. The DNP master driver is to be considered as DNP Subset Level 1 implementation as defined in *DNP V3.00 Subset Definitions Doc Number P009-0IG.SUB*

The DNP slave driver has been developed to test the master driver and may NOT be considered a DNP slave driver as defined in the DNP subset definitions.

Appendix A.3. DNP Objects mapped to FieldServer Data Arrays

DNP objects consist of values and additional information such as quality, control and status bits as well as time information.

The DNP driver allows this additional data to be extracted and mapped into the indicated data array. For example, the DNP master driver can read 10 analog inputs with status flags and put the 10 values in consecutive order in one data array and the 10 status bytes in another data array.

Control of this functionality is achieved by setting up the CSV file correctly. If not specified the DNP driver extracts data values and discards the additional data.

Appendix A.4. Channel Idle, Master & Slave Idle.

The following notes describe the internal architecture of the driver and do not affect the way that the driver is used or configured.

The Driver is implemented using the channel idle. The channel idle function is called in the master mux but must be regarded as processing the channel (independently of the master or slave).

Chan Idle	• processes all incoming bytes, • looks for complete messages, • From a DLL layer point of view parses the message and responds • Signals the master or slave app layer that there is an coming message • Signals master if there is an app layer response • Signals slave if there is an app layer request(read/write) or unsolicited message. • Looks for master or slave app layer signals to process an outgoing message Maintains a list of nodes & node status (in terms of link reset)
Slave Idle	• Looks for signals from chan idle that a message has been received • Parses message from an app layer point of view. • If required sets flags for • Map Descriptor matching, • fetch/store function calls • and/or response function call • Signals Chan idle the outgoing app layer message needs to be processed.
Master Idle	• Looks for signals from chan idle that a message has been received • Processes Map Descriptors and forms read/write messages • Signals Chan idle the outgoing app layer message needs to be processed.

Appendix A.5. DLL Layer Functionality in the Master

The DNP Primer provided by dnp.org describes the DLL layer requests for confirmation as optional and suggests that it is not often employed. Our driver **never** asks for DLL layer confirmations. Thus the DLL layer functions as a mere wrapper/unwrapper layer. It wraps user data with a header and CRC's but does not perform node-node confirmations.

The only DLL layer functions which have been implemented are send and respond with user data and link reset. The slave DNP driver will not respond until a link reset has been performed. The DNP master driver sends a Link Reset request when a Map Descriptor requests data from an un-reset node. The link resetting is performed on a node-node link.

Appendix A.6. App Layer Functionality in the Master

The App layer provides over 40 app layer functions, confirmations and responses and allows for handling of a huge number of data objects.

App Layer Functions	1. Read 2. Write 3. Select 4. Operate 6. Direct Operate with no Ack (limited) 8. Direct Freeze with no Ack (limited) 129. Response 130. Unsolicited. (Slave Driver can parse these messages.)
Internal Indications	The Slave indicates its internal state by appending internal indication bytes to the app layer header of each response. Thus it can report that it is faulty, corrupted or unable to process the request. If it can't find a matching Map Descriptor it sets the internal indication bit used to indicate that the data object parameters specified cannot be parsed. You can configure a Server node to respond with the internal indications bytes that are extracted from a Data Array allowing you to control them. For more information, refer to **Error! Reference source not found.**.
App Layer Qualifier	The app layer contains a Qualifier Byte used to control indexing for data objects. The DNP 3.0 Driver only handles Qualifiers 00, 01, 07, 08, 17, 28. Qualifier 6 is supported with limitations.
App Layer Confirmation	The DNP 3.0 (master) Driver never asks for an App Layer Confirmation. The DNP 3.0 Slave Driver is capable of responding to an app layer request for confirmation (to allow it to process an unsolicited message which may ask for confirmation.)

Appendix A.7. Internal Indications, Object 80 and DNP_II

The driver can store the Internal Indications Bits found in incoming messages and it is possible to control the values of the internal indication bytes sent in responses. In addition, the driver can be configured to respond to the Poll for Object 80 (Internal Indications)

Appendix A.7.1. Incoming Internal Indications Bytes

This driver can expose data from the most recently consumed message and additional diagnostic information using a special Map Descriptor called “DNP_ii” (DNP Internal Indications)

The following example shows the configuration of this Map Descriptor. Only one of these Map Descriptors may be configured per FieldServer.

```
Nodes
Node_Name                ,Protocol
Null_Node                ,DNP

Data_Arrays
Data_Array_Name          ,Data_Format          ,Data_Array_Length
DNP_DIAG                 ,UINT32               ,100

Map_Descriptors
Map_Descriptor_Name      ,Data_Array_Name      ,Node_Name
dnp-ii                   ,DNP_DIAG             ,Null_Node
```

The following data is stored in the Data Array DNP_DIAG.

Array Element	Contents
0	The first byte of the Internal indication reported by a DNP device as found in the most recently received message. Only messages complete enough to warrant parsing will cause this item to be updated. DNP devices only contain internal indication bytes when the message is an application layer response type message. Typically this includes all responses to data queries/writes as well as unsolicited messages.
1	The 2nd byte of the response internal indication.

Bit zero is the least significant bit.

A one (1) in the bit position indicates the described state.

Bit #	Explanation
First Byte	
Bit 0	All stations message received Set when a request is received with the destination address of the all stations address (ffff hexadecimal). Cleared after next response (even if response to global request is required) Used to let the master station know that a Broadcasted message was received by this station.
Bit 1	Class 1 data available Set when data that has been configured as Class 1 data is ready to be sent to the master Master station should request this class data from the Outstation when this bit is set in a

Bit #	Explanation
	response
Bit 2	Class 2 data available Set when data that has been configured as Class 2 data is ready to be sent to the master Master station should request this class data from the Outstation when this bit is set in a response
Bit 3	Class 3 data available Set when data that has been configured as Class 3 data is ready to be sent to the master Master station should request this class data from the Outstation when this bit is set in a response
Bit 4	Time-synchronization required from the master. The master synchronizes the time by writing the Time and Date object to the Outstation. Cleared when the time is set by the master. This bit is also cleared when the master explicitly writes a 0 into this bit of the Internal Indication object of the Outstation.
Bit 5	Set when some or all of the Outstation's digital output points are in the Local state. That is, the Outstation's control outputs are NOT accessible through the DNP protocol. Clear when the Outstation is in the Remote state. That is, the Outstation's control outputs are accessible through the DNP protocol.
Bit 6	Device trouble Set when an abnormal condition exists at the Outstation. The device profile for a given device states the conditions that affect this bit. This should only be used when the state can not be described by a combination of one or more of the other IIN bits.
Bit 7	Device restart Set when the user application at the Outstation restarts. Cleared when the master explicitly Writes a 0 into this bit of the Internal Indications object in the Outstation.
Second Byte	
Bit 0	Function code not implemented
Bit 1	Requested object(s) unknown. The Outstation does not have the specified objects or there are no objects assigned to the requested class. This indication should be used for debugging purposes and usually indicates a mismatch in device profiles or configuration problems.
Bit 2	Parameters in the qualifier, range or data fields are not valid or out of range. This is a catch-all for application request formatting errors. This indication should be used for debugging purposes and usually indicates configuration problems.
Bit 3	Event buffer(s), or other application buffers have overflowed. For example, COS/SOE buffers have overflowed. The master should attempt to recover as much data as possible and indicate to the user that their may be lost data. The appropriate error recovery procedures should be initiated by the user.
Bit 4	Request understood but requested operation is already executing.
Bit 5	Set to indicate that the current configuration in the Outstation is corrupt and that the master application layer should inform the user of this exception. The master may download another configuration to the Outstation. Note that sometimes a corrupt configuration will disable an Outstation, making it impossible to communicate this condition to a master station.

Bit #	Explanation
Bit 6	Reserved for use by agreement, currently always returned as zero (0).
Bit 7	Reserved for use by agreement, currently always returned as zero (0).

Appendix A.7.2. Internal Indications reported in Responses

The Internal Indications (IIN) field is a two-octet field that follows the function code in all responses. When a request cannot be processed due to formatting errors or unavailable data, the IIN is always returned with the appropriate bits set.

Appendix A.7.3. Server_II_Array

This parameter only applies to Server/responding nodes.
If specified the driver validates that the Data Array exists. If it doesn't then Error 78 is printed.
The driver uses the 1st two elements to form the Internal Indications bytes of all normal responses (responses where the driver is able to respond to the poll). The driver does not modify these bytes when building the response but sends the values exactly as found in the Data Array.
If a poll for unknown and/or unsupported objects/devices is received, the driver builds the internal indications bytes itself. They cannot be controlled using this parameter.

Appendix A.7.4. Responses to Polls for Object 80 (Internal Indications)

If a remote client polls for object 80 then the FieldServer needs to be configured with an object 80 Server Map Descriptor.
The length of the Map Descriptor determines the number of bits in the response and hence the number of bytes. A whole number of bytes is always sent so if configured with a length that isn't a multiple of 8 then the driver appends extra zero-state bits to form a whole number of bytes.
Example: This map descriptor responds with 2 bytes of data. 10 bits of the two bytes are extracted from the DA_80 and 6 of them are padded with a state of zero to make a whole number of bits. The driver extracts the bits from consecutive elements of the Data Array. Thus with a length of 10, the driver looks in 10 consecutive elements of the Data Array.

```
Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Address ,Length ,dnpSubType ,dnpDataType ,dnpDataVari ,dnpQualifier
MMBa1               ,0.5s          ,DA_80           ,0                 ,Server   ,Node_A    ,0       ,10      ,value      ,80          ,1           ,Zero
```

Appendix A.8. DNP_Stats

In addition to the standard FieldServer communication statistics described in the FieldServer Configuration Manual the DNP 3.0 Driver can expose some driver statistics by writing to a data array called “DNP_STATS”

The following example shows how this special Map Descriptor can be configured. Only one of these Map Descriptors may be specified per FieldServer.

```
Nodes
Node_Name               ,Protocol           ,Node_ID
Null_Node               ,DNP                ,100

Data_Arrays
Data_Array_Name         ,Data_Format        ,Data_Array_Length
DNP_STATS               ,UINT32             ,100

Map_Descriptors,
Map_Descriptor_Name     ,Data_Array_Name,   ,Node_Name
dnp-stats               ,DNP_STATS,         ,Null_Node
```

The driver stores the following data in the data array EGD_STATS.

Array	Contents
1	DRV_DLL_CLIENT_SENDS_MSG
2	DRV_DLL_CLIENT_SENDS_BYTES
3	DRV_DLL_SERVER_SENDS_MSG
4	DRV_DLL_SERVER_SENDS_BYTES
5	DRV_DLL_CLIENT_RCVS_MSG
6	DRV_DLL_CLIENT_RCVS_BYTES
7	DRV_DLL_SERVER_RCVS_MSG
8	DRV_DLL_SERVER_RCVS_BYTES
9	DNP_UPD_TMO
10	DNP_UPD_PROTO
11	DNP_UPD_NOISE
12	DNP_UPD_CHECK
12	DNP_UPD_STATION
14	DNP_UPD_LENGTH
15	DNP_UPD_EXCEPT
16	DNP_UPD_STREAMING
17	DNP_UPD_FUNCTION
18	DNP_UPD_PREMATURE
19	DNP_UPD_IC_TIMEOUT
20	DNP_UPD_NAK
21	DNP_UPD_OVERRUN

Array	Contents
22	DNP_UPD_HOLD_TIMEOUTS
23	DNP_UPD_NODE_OFFLINE
24	DNP_UPD_NO_SLAVE
25	DNP_UPD_NO_START
26	DNP_UPD_SCADA_BYTES_RECD
27	DNP_UPD_SCADA_BYTES_SENT
28	DNP_UPD_SCADA_MSG_SENT
29	DNP_UPD_SCADA_MSG_RECD
30	DNP_NOT_OK
31	DNP_UPD_RESET_RQD
32	DNP_UPD_CANT_RESET
33	DNP_UPD_NO_NODES
34	DNP_UPD_RESET_FAILED
35	DNP_UPD_MAST_DEBUG_MSG
36	DNP_UPD_MAST_PARSE_ERR
37	DNP_UPD_SLAVE_PARSE_ERR
38	DNP_UPD_MAPD_TOO_SHORT
39	DNP_UPD_TOO_MANY_BYTES4
40	DNP_DIAGNOTIC_GENERATOR
41	DNP_UPD_LINK_RESET_DONE
42	DNP_UPD_LINK_RESET_DONE_BY_MST
43	DNP_UPD_LINK_RESET_DONE_BY_SRV
44	DNP_UPD_LINK_STATUS_STATE

Appendix A.9. DNP 3.0 Data Objects

The DNP 3.0 Driver acting as a Client will produce a single message fragment. A message fragment may contain a maximum of 249 bytes, some of which constitute overhead. The DNP 3.0 driver will panic if the message fragment is too long. Reduce the length and add another Map Descriptor to poll additional items. This limitation does not apply when the DNP 3.0 driver processes a response from a query as the driver can process multi fragment responses.

The list of data objects supported and the functions used to access the objects is defined on the Driver Fact Sheet which may be obtained from FieldServer Technologies. The table is known as a DNP 3.0 Implementation Table.

The table below lists the objects and variations that be used in the Map Descriptors. The DNP 3.0 Driver supports all the objects with some exceptions. The exceptions are noted by indicating the revision

[4] Increments by one each time a response message isn't sent because the number of elements to respond with requires too many bytes to fit in a message.

number of the driver prior to them being supported or by indicating that the object is not supported with the **NS** annotation.

Default Variations are designated with a *. Not all Data Types have a default variation. The default variation will be returned when a Client polls for variation zero (default). Server configurations require a Map Descriptor with variation zero to be defined in the CSV file before the driver can respond with the default variation data.

Object	Variation.	Ex	Description
1	0		Binary Input - All Variations
1	1*		Binary Input
1	2		Binary Input with Status
2	0		Binary Input Change - All Variations
2	1*		Binary Input Change without Time
2	2		Binary Input Change with Time
2	3		Binary Input Change with Relative Time
10	1*		Binary Output
10	2		Binary Output Status
12	0	NS	Control Descriptor - All Variations
12	1*	101a	Control Relay Output Descriptor
12	2	NS	Pattern Control Descriptor
12	3	NS	Pattern Mask
20	0		Binary Counter - All Variations
20	1*		32-Bit Binary Counter
20	2		16-Bit Binary Counter
20	3		32-Bit Delta Counter
20	4		16-Bit Binary Counter
20	5		32-Bit Binary Counter without Flag
20	6		16-Bit Binary Counter without Flag
20	7		32-Bit Delta Counter without Flag
20	8		16-Bit Delta Counter without Flag
21	0		Frozen Counter - All Variations
21	1*		32-Bit Frozen Counter
21	2		16-Bit Frozen Counter
21	3		32-Bit Frozen Delta Counter
21	4		16-Bit Frozen Delta Counter
21	5		32-Bit Frozen Counter with Time of Freeze
21	6		16-Bit Frozen Counter with Time of Freeze
21	7		32-Bit Frozen Delta Counter with Time of Freeze
21	8		16-Bit Frozen Delta Counter with Time of Freeze
21	9		32-Bit Frozen Counter without Flag
21	10		16-Bit Frozen Counter without Flag
21	11		32-Bit Frozen Delta Counter without Flag

Object	Variation.	Ex	Description
21	12		16-Bit Frozen Delta Counter without Flag
22	0	101a	Counter Change Event - All Variations
22	1*	101a	32-Bit Counter Change Event without Time
22	2	101a	16-Bit Counter Change Event without Time
22	3	101a	32-Bit Delta Counter Change Event without Time
22	4	101a	16-Bit Delta Counter Change Event without Time
22	5	101a	32-Bit Counter Change Event with Time
22	6	101a	16-Bit Counter Change Event with Time
22	7	101a	32-Bit Delta Counter Change Event with Time
22	8	101a	16-Bit Delta Counter Change Event with Time
23	0	101a	Frozen Counter Event - All Variations
23	1*	101a	32-Bit Frozen Counter Event without Time
23	2	101a	16-Bit Frozen Counter Event without Time
23	3	101a	32-Bit Frozen Delta Counter Event without Time
23	4	101a	16-Bit Frozen Delta Counter Event without Time
23	5	101a	32-Bit Frozen Counter Event with Time
23	6	101a	16-Bit Frozen Counter Event with Time
23	7	101a	32-Bit Frozen Delta Counter Event with Time
23	8	101a	16-Bit Frozen Delta Counter Event with Time
30	0		Analog Input - All Variations
30	1*		32-Bit Analog Input
30	2		16-Bit Analog Input
30	3		32-Bit Analog Input without Flag
30	4		16-Bit Analog Input without Flag
31	0		Frozen Analog Input - All Variations
31	1*		32-Bit Frozen Analog Input
31	3		32-Bit Frozen Analog Input with Time of Freeze
31	4		16-Bit Frozen Analog Input with Time of Freeze
31	5		32-Bit Frozen Analog Input without Flag
31	6		16-Bit Frozen Analog Input without Flag
32	0		Analog Change Event - All Variations
32	1		32-Bit Analog Change Event without Time
32	2		16-Bit Analog Change Event without Time
32	3		32-Bit Analog Change Event with Time
32	4		16-Bit Analog Change Event with Time
33	0		Frozen Analog Event - All Variations
33	1*		32-Bit Frozen Analog Event without Time
33	2		16-Bit Frozen Analog Event without Time
33	3		32-Bit Frozen Analog Event with Time
33	4		16-Bit Frozen Analog Event with Time
40	0		Analog Output Status - All Variations

Object	Variation.	Ex	Description
40	1*		32-Bit Analog Output Status
40	2		16-Bit Analog Output Status
41	0		Analog Output Descriptor - All Variations
41	1		32-Bit Analog Output Descriptor
41	2		16-Bit Analog Output Descriptor
50	0	101a	Time and Date - All Variations
50	1*	101a	Time and Date
50	2	101a	Time and Date with Interval
51	0	101a	Time and Date CTO - All Variations
51	1*	101a	Time and Date CTO
51	2	101a	Unsynchronized Time and Date CTO
52	0	NS	Time Delay - All Variations
52	1*	1.03b5	Time Delay Coarse
52	2	1.03b5	Time Delay Fine
60	0	1.03b5	
60	1*		Class 0 Data
60	2		Class 1 Data
60	3		Class 2 Data
60	4		Class 3 Data
70	1	NS	File Identifier
80	1	1.03iB	Internal Indications
81	1	NS	Storage Object
82	1	NS	Device Profile
83	1	NS	Private Registration Object
83	2	NS	Private Registration Object Descriptor
90	1	NS	Application Identifier
100	1	NS	Short Floating Point
100	2	NS	Long Floating Point
100	3	NS	Extended Floating Point
101	1	NS	Small Packed Binary-Coded Decimal
101	2	NS	Medium Packed Binary-Coded Decimal
101	3	NS	Large Packed Binary-Coded Decimal

[5] See notes in 0 for function code 23.

Appendix A.10. Controlling the DNP 3.0 Driver's Function Code

When a write Map Descriptor is found by the DNP 3.0 Driver it will use the DNP protocol's write function to write data to the DNP device. If the DNP device's vendor requires that an alternative DNP function be used, use the keyword dnpFunction in the Map Descriptor and specify the function that the vendor requires (provided that it is in the list of functions implemented by this driver.)

Example: To set a Control Relay Output Descriptor's state in a SEL-351A relay you cannot use a write function. The vendor requires you use functions 3, 4, 5 or 6. Select the appropriate function and specify it as the value of the dnpFunction parameter.
In some cases the DNP 3.0 driver will change the default function automatically. This will be reported in the error log.
The dnpFunction parameters may have any legal DNP protocol function. The function must be specified as a decimal value.

Function	Value for dnpFunction	Client Supports	Server Supports
Confirm	0	Yes	Yes
Read	1	Yes	Yes
Write	2	Yes	Yes
Select	3	Yes6	Yes6
Operate	4	Yes6	Yes6
Direct Operate	5	Yes6	Yes6
Direct Operation with no Ack	6	Yes6	Yes6
Immediate freeze	7		
Immediate freeze with no Ack	8	Limited*	
Freeze Clear	9		
Freeze Clear with no Ack	10	Limited*	
Freeze Time	11		
Freeze Time with No Ack	12		
Cold Restart	13		
Warm Restart	14		
Delay Measurement	23	Yes	Yes

When the Function #23 (Delay Measurement) is used the following two notes apply:

1. When the driver is configured as a Client, the configuration must also specify the *dnpDataType* parameter set to Object 52 (Time Delay Objects) and the *dnpDataVari* parameter to variation 2 (Time Delay Fine). The driver reads the single object from the Server. The address parameter is ignored.

2. When the driver is configured to serve a response to a poll with application function #23 then the configuration must have a Server Map Descriptor with *dnpDataType* parameter set to Object 52 (Time Delay Objects) and *dnpDataVari* parameter set to variation 2 (Time Delay Fine). The address parameter is ignored as the poll is for a single object. The driver responds by serving

[6] There are limitations to the Select and Operate Functionality. Refer to Appendix A.19 for more information.

data from the Data Array element specified on the Server Map Descriptor – most DNP 3.0 RTU devices respond by reporting the number of milliseconds between receiving the 1st bit of the poll and sending the 1st bit of the response.

Appendix A.11. Controlling the DNP 3.0 Driver's Qualifier

The protocol uses a qualifier in the application layer part of a message to indicate how data must be packed in the message. The DNP 3.0 Driver as a Client will default to use a Qualifier of 1. The qualifier is used internally by the driver and is of no concern unless the DNP protocol has been implemented to require the use of qualifiers other then 1. In this case it is possible to change the driver's default value by specifying the dnpQualifier parameter. For example, to address a Control Relay Output Descriptor object in a Multlin 489 Generator management relay, the vendor data sheet indicates that only qualifiers 17 and 28 may be used.

When the driver is configured as a Server then the dnpQualifier parameter specified in the CSV file is not used. The qualifier of the requesting poll is used in interpreting the request and forming the response. The driver accepts poll's that have the DNP qualifier set to 6 which effectively requests the Server to send back all objects of the type requested. The driver's support for this qualifier is limited by the length of the response message. The response must fit in one 255 byte message. The number of objects contained in the response message (and hence the length of the response) is controlled by the length parameter. We have found lengths in the upper range 10 to 30 successful.

As a Server, if configured to serve 'changed' data the driver may choose the qualifier itself as the data object may be non sequential. See 5.3.8.

Appendix A.12. FieldServer DNP Node Number

The FieldServer's DNP node number is set in the CSV file. In the following example the node is set to 22.

```
FieldServer
Title          ,System_Node_ID
B1             ,22
```

Appendix A.13. DnpSubType

DNP objects often contain more than one element of information, e.g. Object 30, variation 1 is a 32-bit analog input. When the DNP device is polled for data for this object the device returns a data structure which contains a 32 bit value for the input and an 8 bit status byte indicating the input's quality. Alternatively the FieldServer defaults to the ***value*** subtype.

By using the ***value*** or ***flags*** in the field for dnpSubType you could have the FieldServer extract the value or the status byte and place them in the data array associated with the Map Descriptor. In this example it would have made no sense to try and extract a time as there is no time field associated with object 30, variation 1

Permitted Values	Description
Value	The driver extracts the value of the object being read.
Flags	The driver extracts the quality/status byte of the object.
Time1	The driver extracts the time field from the object.
Time2	The driver extracts a second time field from the object if the object has more than one.
Combo (NB - only valid with Object 12, Variation 1; Object 41, Variation 1 & 2)	Used with a write Map Descriptor with the dnpFunction set to 5 (Direct Operation with no Ack). The driver uses multiple consecutive elements from the data array to build the write command. When used with Object 12 Variation 1. The 1st array element is used as a byte to fill in the control code field. The 2nd array element is used as a byte to fill in the count field. The 3rd array element is used as a UINT32 to fill in the on time field. The 4th array element is used as a UINT32 to fill in the off time field. The status field is always set to zero. When used with Object 41 Variation 1 & 2 The 1st array element is used as a INT32 (variation 1) or INT16 (variation2) to fill in the requested value field. The 2nd Array Element is used as a byte to fill in the control status field. (See the definition of object 12 for a description of this field.)

Appendix A.14. Communication Stats

The driver counts bytes on the connection and messages on the Map Descriptors. Thus if a Map Descriptor is used to generate a poll then the transmit messages increment and when a response is received, its received messages will increase. The byte count of these messages will be counted on the connection and not on the Map Descriptor. The connection also counts messages and bytes that form connection related messages such as link resets, confirms, acks, naks….

Appendix A.15. Link Reset

The Link Reset message forms part of the Data Link layer of the DNP 3.0 protocol. It is used to establish and check a connection. When configured as a Client, the driver sends a Link Reset and waits for the Server to respond before starting data transfer. When configured as a Server, the driver will not respond to messages until the remote Client has sent a Link Reset. The following example configuration will override this default behavior.

Example

```
//  Server Side Connections

Connections
Port        ,Baud       ,Parity     ,Protocol   ,Application
P8          ,9600       ,None       ,DNP        ,NoLink
```

Appendix A.16. Controlling DA Offsets

Versions of the driver prior to 1.02a used a method to calculate DA offsets that was not consistent with the general FieldServer model. This was corrected in versions 1.02a and later. To retain the old offset calculation, specify the 'Application' parameter on the connection.

Use either 'OriginalStyle' or 'OrigStyle-NoLink' (to overwrite Link Reset).

Example

```
//  Server Side Connections
Connections
Port        ,Baud       ,Parity     ,Protocol   ,Application
P8          ,9600       ,None       ,DNP        ,OriginalStyle
```

Appendix A.17. dnpIndexStyle

Normally the Server responds to requests for data using the same qualifier as the poll. Use this parameter on a Server Map Descriptor to override the indexStyle of the qualifier in the response. Refer to Section 5.3.6 for an example.

Supported

1 objects are prefixed with a one byte index

2 objects are prefixed with a two byte index

Unsupported

3 objects are prefixed with a four byte index

4 objects are prefixed with a one byte object size

5 objects are prefixed with a two byte object size

6 objects are prefixed with a three byte object size

Others Unknown.

Appendix A.18. Real Time Clock Synchronization

The FieldServer has a real time clock. ProtoCessors have a software clock. If using a ProtoCessor, it is advisable to update the clock on the device periodically.

Remote update of Real Time Clock – Server configuration

Regardless of the platform in use, it is possible to have the FieldServer set its clock using the DNP 3.0 protocol when the driver is configured as a Server. The following actions are required:

- Configure the driver to perform remote updates (Node parameter 'Real_Time_Clock_Control' = 'Update-from-Remote'.
- Set update frequency (Connection parameter 'RTC_Update_Rqst_Interval', default -2000. Value is specified in milliseconds.)
- Have the Client send the time using Object 50 variation 1 (Time & Date CTO: Absolute Time) to the Server. When the Server receives this time it will store the value in a Data Array, as well as use it to update the real time clock

Example

```
//    Server Side Connections
Ports
port                ,Baud               ,Protocol           ,RTC_Update_Rqst_Interval
P1                  ,2400               ,DNP                ,123
```

```
//    Server Side Nodes
Nodes
Node_Name           ,Node_ID            ,Protocol           ,Real_Time_Clock_Control
Node_A              ,11                 ,DNP                ,Update-from-Remote
```

```
Map_Descriptors
Map_Descriptor_Name ,Data_Array_Name    ,Data_Array_Offset  ,Function ,Node_Name ,Address ,Length ,dnpDataType ,dnpDataVari
Store_Time          ,DA_TIME            ,0                  ,Server   ,Node_A    ,0       ,50      ,50          ,1
```

In this example, the Server has been configured to request a time update every 123 milliseconds and the request will be made by notifying the remote device that an update is required. The Server does this by setting a flag in the response internal indications bits. The default update is 2 seconds. If using the default, omit the specification of the 'RTC_Update_Rqst_Interval' parameter.

Remote update of Real Time Clock – Client configuration

If using the DNP 3.0 driver as a Client and you want to configure the driver to respond to Server requests for real time clock updates then:

- Create a Data Array named 'DNP3_RTC_NODEx' where x is the Server node station address. The format for the Data Array should be UINT32.
- Ensure that DA is constantly updated by value you wish to set the real time clock to. The format of the value is a number that contains the number of seconds since Jan 1 1970.

The driver will auto create a Map Descriptor to write the time to the Server using the Data Array. If the Data Array doesn't exist then the driver will print an error message.

Limitations:

- Test the accuracy of the ProtoCessor software clock to determine its limitations. The accuracy may be dependent on environmental conditions.
- The clock has a precision of seconds.

Appendix A.19. Select and Operate

As a Client

The DNP 3.0 Driver Client can send select, operate and direct operate messages to a Server device. The commands can be sent to Analog Output Blocks, Relay Control Blocks and Pattern Control Blocks. The responses from these commands contain status information on whether the command was accepted or rejected. The driver can store this status information so that is it available to another protocol.

When using these commands it is important to remember that the driver is not a DNP 3.0 device and is only capable of transferring data from one protocol to another. The driver does not understand the context of these commands. For example, the driver can send the select command but will not resend the command if an operate command failed because the select arm timer had expired. Another example, is when you send an operate command to a control relay, the driver sends the relay a block of data that is extracted from a Data Array. It is the content of this block that tells the relay what to do. Because the driver doesn't understand the command but simply sends a block of data it is necessary to ensure the block of data has been formatted correctly.

The driver does not check the echoed data objects contained in the response. The DNP 3.0 specification requires that a DNP 3.0 device should.

Analog Output Block

Depending on the variation selected the driver sends a 16 or 32 bit signed value with a status byte (Status byte sent is always equal to zero).

If the active Map Descriptor used to send a select or operate command has the DA_Float_Name parameter specified with the name of a Data Array then the driver stores the status byte extracted from the response message in this secondary Data Array at the same offset specified with the "Data_Array_Offset" parameter.

Control Relay Block (Object=12)

The following 5 fields are sent with the command to operate such an object:

Field	Description
Control Code	This field indicates the control function to perform. The applicability of this code will depend on the type of hardware used in the end device.
Count	This field indicates the number of times that the control operation should be performed in succession.
On-time	This field specifies the amount of time the digital output is to be turned on (may not apply to all control types).
Off-time	This field specifies the amount of time the digital output is to be turned off (may not apply to all control types).
Status Byte	This is always sent with a value of zero.

By default the driver grabs one item from the Data Array specified on the Map Descriptor. It uses this value for the control code, sets the count field to 1 and sets the time and status fields to zero.

If the “dnpSubType” is specified with a value of “Combo” then the driver will extract 4 consecutive values from the Data Array specified on the Map Descriptor. These values are used to complete the ‘Control Code’, ‘Count’, ‘On-Time’ and ‘Off-Time’ fields in the message.

If the active Map Descriptor used to send a select or operate command has the “DA_Float_Name” parameter specified with the Name of a Data Array then the driver stores the status byte extracted from the response message in this secondary Data Array at the same offset specified with “Data_Array_Offset” parameter. If the “DA_Bit_Name” parameter is specified then the driver stores the control code extracted from the response and if the “DA_Byte_Name” is specified then the driver stores the count field extracted from the response.

Pattern Control Block

Object 12 variations 2 & 3 are not supported by the driver.

As a Server

The DNP 3.0Driver Server can accept and process Select and Operate commands sent by a remote Client with limitations as discussed in the notes below.

The commands can be sent to Analog Output Blocks, Relay Control Blocks and Pattern Control Blocks. The driver executes or rejects the commands on these data objects, returning a response which contains an echo of the poll with the status byte of each data object modified to report the success or failure.

The driver is intended to transfer data from one protocol to another. The driver does not make the FieldServer behave like a typical DNP 3.0 device such as a protective relay. For these reasons the driver cannot implement all the actions implied by the Select and Operate commands. For example: The driver cannot queue commands. The driver cannot use the on and off time fields. The driver only understands the ‘Latch On and “Latch Off’ command codes. The driver cannot understand the attribute bits of the control code field such as the clear and reset attribute flags.

All Commands

When one of the select, operate and direct operate commands are received, the driver sends a response which contains internal indication status bytes as well as en echo of the data objects being commanded. Each data object that can be commanded with one of the commands has a status byte field trailing the block of data sent with the object. The Server modifies the status byte that is echoed back.

The driver sends 1 of 4 possible status byte responses.

0 Command Accepted

2 Command rejected because object was not previously selected.

4 Command rejected because the command code is not supported by the driver.

6 Command Rejected, Hardware problem. The driver sends this response when the driver is not able to process the command correctly. It does not mean that the FieldServer hardware is malfunctioning.

Select Command

This command is used to select data objects for an 'Operate' Command. If possible the driver marks the data objects as selected when the select command is received. If not possible, the driver assumes the objects are selected at all times. Many DNP 3.0 devices implement an 'Arm' timer that clears the select flag when it times out. This driver does not implement this timer but relies on a remote device using another protocol to implement the 'Arm' timer if it so requires. The remote device can clear the select flag by writing a zero to the Data Array element at any time.

If the Server Map Descriptor used to define the Server data object to be selected/operated has the "DA_Bit_Name" parameter specified, the driver uses this Data Array location to set the select flag by setting the value to '1'. If the secondary Data Array is not specified then the select command is accepted and the point is considered to be permanently selected.

The driver sends responses with the status byte modified to report the success/failure of the command.

Operate Command

Once selected, data objects can be operated. If the secondary Data Array is specified using the "DA_Bit_Name" parameter, the driver looks in the Data Array and if the value is non-zero the point is considered to be selected and the operation can proceed. If the value is zero, the driver rejects the command. If the secondary Data Array is not specified, the port is considered to be permanently selected.

The driver supports the following Control Codes

Code	Operation	Supported
0	Null	No
1	Pulse On	No
2	Pulse Off	No
3	Latch On	Yes
4	Latch Off	Yes
5-15	Undefined	N/A

'Direct Operate' and 'Direct Operate with No Acknowledgement' Commands

The driver does not check if the data object had previously been selected. The supported Control Codes are described above.

Analog Output Block

When a 'Select' command is received the driver sets the select flag in a secondary Data Array defined by the "DA_Bit_Name" parameter, if available. When any of the operate commands are received the driver stores the value sent with the command in the Data Array defined on the Map Descriptor.

Control Relay Block (Object=12)

If the Control Code is supported the driver does the required action.

The driver supports the following Control Codes

Code	Operation	Supported
0	Null	Not Supported
1	Pulse On	Not Supported
2	Pulse Off	Not Supported
3	Latch On	Driver sets Data Array value to ‘1’
4	Latch Off	Driver sets Data Array value to ‘0’
5-15	Undefined	N/A

If the “DA_Bit_Name” parameter is specified then the driver stores the control code as extracted from the command and if the “DA_Byte_Name” is specified then the driver stores the count field, extracted from the command message.

Pattern Control Block

Object 12 variations 2 & 3 are not supported by the driver.

Appendix A.20. Multiple requests in a single poll.

To provide level 2 compliance a DNP 3.0 device must support the ability to serve responses to single polls that contain requests for multiple object types. Outstations commonly use this technique to send a single message requesting class 1, 2 and 3 data. From version 1.03j onwards the driver acting as a client can send requests which contain multiple objects.

To have the driver request multiple object types in a single poll it is necessary to associate multiple Map Descriptors for this common purpose. This is done by specifying the 'dnpMultiMsg' parameter in the Map Descriptor. Assign the same positive whole number to all Map Descriptors to be associated. When the driver processes an active Map Descriptor with a non zero dnpMultiMsg parameter value it locates other Map Descriptors with he same dnpMultiMsg parameter value and uses all of these to form the poll.

When configuring the Client side Map Descriptors for this purpose, only one of the associated Map Descriptors may be set to active. The other Map Descriptors must be set as Server/Passive.

In the example below the driver finds the Map Descriptor called Class0. It sees that dnpMultiMsg is non zero and has a value of 1. It searches through the remaining Map Descriptors to find any other with dnpMultiMsg =1.and finds the Map Descriptor called Class1. Now the driver sends a single message that requests class 0 and class 1 info.

```
Map_Descriptors
Map_Descriptor_Name ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Address ,Length ,dnpDataType ,dnpDataVari ,dnpAssociate ,Scan_Interval ,dnpMultiMsg
3                   ,DA_Class0       ,0                 ,Rdbc     ,Node_A    ,0       ,1       ,60          ,1           ,1           ,2.0s          ,1
Class1              ,DA_Class1       ,0                 ,Server   ,Node_A    ,0       ,1       ,60          ,2           ,1           ,2.0s          ,1
```

If the driver is configured as a Server then no action is required. Simply ensure that that there is a Server Map Descriptor for each object requested. No special actions are required to configure the Server to respond to requests for multiple object types.

Appendix B. DRIVER ERROR MESSAGES

Error Message	Explanation
DNP:#1 Err. Unknown object=%d variation=%d for resp.	The driver is polling for or responding to a poll for a data object and variation that is not supported. See Appendix A.9 for a table of supported objects.
DNP:#2 Err. Unknown object=%d variation=%d for store.	
DNP:#3 FYI. No nodes to process.	No DNP 3.0 nodes have been defined in the configuration. The configuration file is probably invalid. 7.
DNP:#4 Err. Link Reset node %d failed. (Send a Link Reset Rqst before polling.)	A Client must send a 'Link Reset Request' message to this driver before it can respond to a poll.
DNP:#5 FYI. Slave=%d happy with link. No reset Rqst.	Many DNP 3.0 Servers time out the link and the connection must be reestablished with a 'Reset Link Rqst' message. This driver doesn't do this. This message is printed when the link has timed out and a poll is received. The message indicates that this behavior is ok.
DNP:#6 FYI. Node=%d requires a reset before it can respond.	A Client must send a 'Link Reset Request' message to this driver before it can respond to a poll.
DNP:#7. Err. Too many polls with no reset request.	When the driver receives too many polls without a link reset request then this message is printed and the driver panics.
DNP:#8 Err. Cant reset node=%d (Max=%d)	The driver cannot reset the link for the node reported in the message because the node number is too large. Change the Client to poll for a smaller node number.7
DNP:#9 Err. Unknown dnpSubType=<%s>	Valid values are provided in Appendix A.3. The driver uses 'value' as default if the keyword is not recognized.
DNP:#11 FYI. Master Node Address=%d	The driver is reporting the Node_ID of the Client node. This message is for information only.
DNP:#12 FYI. DEBUG Message being sent.	You should never see this message. Check the configuration file for an illegal DNPDatatype.7

[7] Edit the CSV file, download to the FieldServer and reset the FieldServer.

Error Message	Explanation
DNP:#13 FYI. Write function overwritten. MD=<%s>	This message is for information only. When writing to a DNP device the message contains a function code which may be overwritten by specifying the DNPfunction in the CSV file. When DNPfunction has not been specified, the default function is used and this message is printed. When writing to a relay output the driver uses function code=5 (Direct act with no ack). When writing to a Counter the driver uses function code=8 (Immed freeze with no ack).
DNP:#14 FYI. Length truncated to 255 for MD=<%s>	When using DNP qualifier 17, the maximum value for Map Descriptor Length is 255. The driver has truncated the request 7
DNP:#15a Err. Cant understand index portion of qualifier. Qual=0x%02x for MD=<%s>	The qualifier is invalid. Appendix A.11 has a list of valid Qualifiers.7
DNP:#16 Err. Previous Poll not completed	An attempt was made to start a new poll before the previous one completed. A poll might consist of multiple transactions of multiple fragments and therefore could take several seconds. Take a log and send with the problem description and the configuration file to FST Tech support.
DNP:#17. Err. Previous Write not completed	
DNP:#18 FYI. Cmd completed but no (event) data to store.	These messages are printed when the polls for data return with no data. This can be normal if there are no events/ diagnostic data in the remote device. This message is for information only.
DNP:#18 FYI. Cmd completed but no (diag) data to store.	
DNP:#20 FYI. Node Indicates that Event Buffers Have Overflowed.	This message is for information only. It is printed when a remote device reports that its event buffers have overflowed.
DNP:#21 FYI. Length=%d invalid. Setting to %d.	
DNP:#23 FYI. Store rqsted but no Data.	A poll for data contained a response with no data. This can be valid, e.g. you request Change Data and there isn't any.
DNP:#25 FYI. Slave Cant use Qual=6. for Md=<%s>	You cannot use a qualifier of 6 when configuring a Server. See Appendix A.11 for a list of valid qualifiers. 8

[8] Edit the CSV file, download to the FieldServer and reset the FieldServer.

Error Message	Explanation
DNP:#27 Err. Slave: Too many bytes=%s to send. Reduce message length	Change the Map Descriptor length so that the message contains fewer data bytes. 8
DNP:#29 Err. Cant process this qualifier (%d)	The driver does not support this qualifier when processing response from a remote DNP device. Configure the device to use a different qualifier.8
DNP:#30 FYI. Data to store but no mapDesc found.	A message has been received from a remote DNP device but the driver has not been configured to store this data.
DNP:#31 Resp. data contained obj=%d vari=%d. MD rqd for storage.	Some responses (Class Data) contain multiple data types. The driver could not find a Map Descriptor to store some of the data. Refer to Section 4.4.1 to correct the problem. 8
DNP:#32 Err. Cant store for index style %d	The driver can only store data from messages where the data elements are indexed in a supported style. Take a log and send with your configuration file to FST Tech Support.
DNP:#33 Err. Qty items + array offset > end array. md=<%s> Cant store all. qty/max= %d/%d	You need to make the data array longer (and/or the length of the Map Descriptor.) 8
DNP:#35 Err. Relay Output. Status data array too short.	The Data Array designated with e DA_Byte_Name parameter in the CSV file is too short. 8
DNP:#36 Err. Relay Output. Status data array too short.	
DNP:#37 Err. Relay Output. Bridge cant implement action code	The driver only supports ‘Latch On’ and ‘Latch Off’ and Relay output actions. Reconfigure the remote device to use a different relay action.
DNP:#38 Err. Storage Method Unknown. (%d)	Take a log and send the log and configuration file to FST Tech Support.
DNP:#40 Err. Abandoned data store. Too many data object types.	The DNP 3.0 driver can process a max of 20 different data objects when storing class data. Reconfigure the remote device for fewer data objects in the class requested.
DNP:#41 Err. MD=<dnp-ii> is too short. Rqd=%d	9
DNP:#41 Err. MD=<dnp-stats> is too short. Rqd=%d	

Error Message	Explanation
DNP:#42 FYI. You could have used a MD called <dnp-ii> to expose diagnostic info.	**Error! Reference source not found.** and 0 contain additional information.
DNP:#42 FYI. You could have used a MD called <dnp-stats> to expose diagnostic info.	
DNP:#43a Err. Diagnostic #3. Call Support.	If any variation of MSG #43 is printed, take a log and call Tech support.
DNP3:#44 FYI. Link Reset Suppressed. Port=%d	The configuration file has suppressed the requirement for link reset. If this corresponds to your expectation then you can safely ignore this message, If not, see Appendix A.15
DNP:#45 Err. Dest Address=%d=Broadcast Address." , dest_station DNP:#45 Err. Perhaps the System_Station_Address has not been specified."	If the destination address is 0xFFFF (=65535) then this message is printed. The address 0xFFFF is the broadcast address and the driver cannot process broadcasts. Configure the node which sent the broadcast message to send a specific message to the FieldServer. 9 We have also seen that this message gets printed when the 'System_Station_Address' has not been specified. Appendix A.12 has more information.
DNP:#46 Err. Server received a poll for unknown node=%d	A message was received by the Server side for a Node that has not been defined (as a DNP node). Either this message wasn't intended for the FieldServer or the configuration requires review. You may need to add a new Server side node and Server side Map Descriptors to solve this problem.
DNP:#49 Err. Cant process func=0x%x from a responder DNP:#50 Err. Cant process func=0x%x from an initiator	The driver cannot process the function code indicated when the message comes from a responder. If this message occurs rarely then assume it is the result of an occasional corrupt message. If you are concerned or if it occurs frequently then take a log and contact Tech Support.
DNP:#51 Err. Node_ID=%d Valid Range=0-65535	Illegal Node_ID used. The Node ID refers to the DNP 3.0 Station Address. Refer to sections **Error! Reference source not found.** and 0.

[9] Edit the CSV file, download to the FieldServer and reset the FieldServer.

Error Message	Explanation
DNP3:#52a FYI. OriginalStyle DA offset use.	The message reports that the driver has been set to a mode where it will use the DA offsetting style present in the driver prior to version 1.02a. If this is consistent with your configuration, the message may be ignored.
DNP3:#53 FYI. Mast App Parse - NO DATA, returning early	A response to a poll contained no data. This is possible when, for example, you poll for event or change data and no events/changes have occurred since the last poll. This message is printed for information purposes only
DNP3:#53b FYI. Class Rqst returned no data.	As per #53 but pertains exclusively to polls for class data.
DNP:#54 Err. App Layer Func=%#x not recognized.	A message was received by the Server side and the application layer function code isn't supported by this driver. You could reconfigure the Client not to send this message,
DNP:#55 Err. To serve class data set the DNPAssoc value non-zero	When you create MD's to serve Class Data the dnpAssoc field must have non-zero values. More information is provided in section 5.3.59
DNP:#56 Err. No MDs associated with the class MD=%s	To serve Class data you need a Map Descriptor matching the class data request and a number of associated Map Descriptors to tell the driver what data forms part of the class data set. More information is provided in section 5.3.59
DNP:#57 FYI. You could have used a MD called <dnp-stats> to expose diagnostic info.	**Error! Reference source not found.** and 0 contain additional information.
DNP3:#58 Err. Response Dest=%d != Master=%d	The driver found that a response message was addressed to a node different from the polling node. Take a log and report to Tech support.
DNP3:#59 Err. Response Unknown Src_Id=%d	The response message originated from a node different to the polled node. If the Node_ID reported is not in the list of nodes in your configuration, take a log and report the problem to Tech Support. If this message is printed rarely, it may have resulted from an occasional corrupt message.
DNP3:#60 Msg Frame abandoned. Frames out of seq. Exp=%d Rcvd=%d. TP=%#d	Some DNP messages contain too much data to be sent in one message. Such messages are split into multiple frames and reassembled on the receiving side. Each frame has a sequence number. If frames are received out of sequence this message is printed. Typically, once one frame is found out of sequence the remaining frames may be flagged as errors too. A consequence of this message is the loss of the complete message. If this message is printed rarely, you could assume that it is a consequence of occasional corruption. If it is printed often then take a log and call Tech support.

Error Message	Explanation
DNP3:#61 FYI. Read func changed Now=%s. MD=<%s>	When the DNPfunction parameter is omitted on a Map Descriptor used to read data the driver assumes that the intended DNPfunction is 'read'. However in some case the driver knows that for certain Objects the function must be changed and does this automatically. This message gives a heads. No action is required from you.
DNP3:#62 Err. Rejected msg with multiple polls.	The Server side of the driver can only process messages which poll for one class of data at a time. Re-configure your Client and try again. This message is obsolete. Only driver version 1.03i and older produce the message. Read Appendix A.20 for more info.
DNP3:#64 Err. To serve class data set the DNPAssoc value non-zero	To serve class data the parameter dnpAssociate must be specified on all associated Map Descriptors and the value of the parameter must be non-zero.10
DNP3:#65 Err. There are no MD associated with the class MD=%s	Refer to Section 5.3.5. 10
DNP3:#63a Err. Cant open %s for poll from log	If any of these messages are printed call tech support. An internal diagnostic specific to QA testing has been activated. It is possible for the driver to send a message that is found in a log file instead of the configured poll to test the Server side of the driver. Script S4085 provides an example. The Client side MD name must have '.log in the name and the driver opens that file and sends the 1st line (only) as a single message. The line is expected to have hex bytes delimited by space or square brackets.
DNP3:#63b Err. Cant read 1st line of %s	
DNP3:#63c FYI. Sending message from log file=%s	
DNP3:#66 Err. Index Style=%d not supported.	The Server side of the driver does not support the indicated index_style. Refer to Appendix A.17. Reconfigure your Client to use a different qualifier and try again10
DNP3:#67 FYI Serve 'changed' data only. Class=%d	This message confirms that the Server has been configured to serve 'changed' data only. Thus, only data whose value has changed by some dead band since the last poll for class data will be served. You can safely ignore this message if it confirms your expectations. If not then review your configuration.
DNP3:#68 FYI. Class Data Served with Qualifier=%#x	When the Server serves class data and when configured to serve only 'changed' data then the Server may change the qualifier of the response from the default or configured qualifier because the points being served in the response may not be sequential. You may ignore this message as it does not report an error. It is printed to draw your attention to the fact that the master should be configured to receive a response that uses a different qualifier.

[10] Edit the CSV file, download to the FieldServer and reset the FieldServer.

Error Message	Explanation
DNP3:#69 FYI. Server signals Client=%d to time synch	Each time that the Server sets the internal indication bit to signal the master station that a time update is required this message is printed. You can safely ignore it if it confirms your expectations. Refer to Appendix A.18.
DNP3:#70 Err No COV DeadBand for DA=%s Off=%d	The Server has been configured to serve 'changed' data only. In reviewing the data, the driver could not find the dead band needed to make a decision about whether the data has changed or not. Refer to Section 5.3.10 for an example showing configuration requirements. 11
"DNP3:#71 FYI. Client observes: Server rqsts time synch	Each time that the Server sets bit 4 of byte 1 of the internal indication field in the response message the Client driver will print this message. It alerts you to the fact that the Server device expects to be sent the time and to have the master reset the flag. Refer to Appendix A.18 for more information on how to configure the master to respond to this request. If you don't care about the Server's time then ignore this message without consequence.
DNP3:#72 Err. Heading not equal to keywords	The driver attempted to auto-create a Map Descriptor to send the time to a Server device that has requested a Real Time Clock Update. The attempt failed and the time was not sent. To resolve this error refer to Appendix A.18 and review your configuration. If you cannot see a problem, capture a log and send the log with the configuration to FieldServer Tech Support.
DNP3:#73 FYI. Time Synch MD Created	You can safely ignore this message. It confirms that the Client side of the driver, saw a request for a time update and auto-created a MD to send the time to the Server device. Read Appendix A.19 for more information.
DNP3:#74 Err. Ana Output Blk. No Operate because no select. Obj=%d"	Analog Output Blocks and Relay Blocks must be selected before they can be operated. The driver needs a place to store the select flag. This is done using a Secondary Data Array specified with the DA_Bit_Name parameter. If not specified then the point is assumed to be selected all the time. Errors #75 and #77 can safely be ignored. If a point has secondary storage defined to store the select flag then the driver checks the select flag has been set when an operate command has been received. If not set then the message #74 or #76 is printed. These errors can be avoided by having the remote master send a select command first. Refer to Appendix A.19 for more information.
DNP3:#75 FYI. Relay Output. No place to store select. Obj=%d"	
DNP3:#76 Err. Relay Output. No Operate because no select. Obj=%d	

[11] Edit the CSV file, download to the FieldServer and reset the FieldServer.

Error Message	Explanation
DNP3:#77 FYI. Ana Output Blk. No place to store select. Obj=%d"	
DNP3:#78 Err. Node_ID=%d II Array=%s doesn't exist.	The configuration contains the Node parameter 'Server_II_Array' but when the driver looked for the Data Array specified it could not find it. The Data Array must be defined before this parameter is used. If the Data Array cannot be found then the driver responds as if it had not been specified at all.
DNP3:#79 FYI. Node=%d Using DA=%s for II	The driver prints this message once and then suppresses it. It draws your attention to the fact that one or more Server nodes has the node parameter 'Server_II_Array' specified and that the Internal Indications bytes in the response will not be built by the driver but will rather be extracted from two consecutive data array elements. Offsets 0 and 1 are always used. If this message is consistent with your expectations then ignore it otherwise review your configuration
DNP3:#80 FYI. Time Synch requires DA=%s	To auto-create a Map Descriptor to send the time to the Server device the driver required the configuration to have a Data Array called 'DNP3_RTC_NODEx' where x is the Server device station address. Refer to Appendix A.18 for more information11
DNP3:#85 Err. Fragment acknowledgements not supported.	The driver does not support message fragment acknowledgements. Please configure your client software not to request them. This message is printed each time we receive a message with CON flag set in the Application control flag. "CON If set to one (1) in a received message, indicates the sending application is expecting a confirmation from the receiving application of the reception of the fragment. An application function code zero (0) is used in the confirmation message." Extract from the DNP3 spec.
DNP3:#86 Err. Index style=%d is not supported. Support 0,1,2. Reformat your request qualifier.	The driver supports qualifiers with an index style of 0, 1 or 2. Index styles 3,4,5,6 are not supported. The only way to work around this is to reconfigure the client software to poll using different qualifiers.